An Abacus Turn Model for Time/Space-Efficient Reconfigurable Routing

Binzhang Fu, Yinhe Han, Jun Ma, Huawei Li, and Xiaowei Li
[a)]Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b)]Graduate University of Chinese Academy of Sciences, Beijing, China
{fubinzhang, yinhes, majun, lihuawei, lxw}@ict.ac.cn

ABSTRACT

Applications' traffic tends to be bursty and the location of hot-spot nodes moves as time goes by. This will significantly aggregate the blocking problem of wormhole-routed Network-on-Chip (NoC). Most of state-of-the-art traffic balancing solutions are based on fully adaptive routing algorithms which may introduce large time/space overhead to routers. Partially adaptive routing algorithms, on the other hand, are time/space efficient, but lack of even or sufficient routing adaptiveness. Reconfigurable routing algorithms could provide on-demand routing adaptiveness for reducing blocking, but most of them are off-line solutions due to the lack of a practical model to dynamically generate deadlock-free routing algorithms.

In this paper, we propose the abacus-turn-model (AbTM) for designing time/space-efficient reconfigurable wormhole routing algorithms. Unlike the original turn model, AbTM exploits dynamic communication patterns in applications to reduce the routing latency and chip area requirements. We apply forbidden turns dynamically to preserve deadlock-free operations. Our AbTM routing architecture has two distinct advantages: First, the AbTM leads to a new router architecture without adding virtual channels and routing table. This reconfigurable architecture updates the routing path once the communication pattern changes, and always provides full adaptiveness to hot-spot directions to reduce network blocking. Secondly, the reconfiguration scheme has a good scalability because all operations are carried out between neighbors. We demonstrate these advantages through extensive simulation experiments. The experimental results are indeed encouraging and prove its applicability with scalable performance in large-scale NoC applications.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Interconnection architectures; C.1.4 [**Parallel Architectures**]: Distributed architectures

ISCA'11, June 4–8, 2011, San Jose, California, USA.

General Terms

Algorithms, Design, Theory

Keywords

Network-on-Chip (NoC), virtual channel (VC), wormhole, turn model, reconfigurable routing, deadlock-free routing, adaptive routing, load balancing

1. INTRODUCTION

With today's micro-architecture technology, tens of cores are possible on a single chip multiprocessor (CMP), such as Intel's terascale processor [1] and Tilera's TILE64 processor [2]. Recently, Tilera corporation has announced its plan to double the processor cores on a single chip every two years [1].

To efficiently interconnect such a high number of elements, Network-on-Chip (NoC) has been widely accepted as the replacement to the shared buses or dedicated wires [1, 2, 3].

To reduce packet latency and buffer requirement, wormhole routing has become one of main paradigms [4]. In wormhole routing, packet is divided into fixed-sized flow control digits (flits), including a head flit, several payload flits, and a tail flit. The head flit is responsible for establishing the routing path, which is followed by all succeeding flits and released by the tail flit. Since a packet could move across several nodes simultaneously, very short network delay is incurred. However, when the head flit is blocked, all flits must stay alone the path. This, together with the fact that applications' traffic tends to be very bursty [5], significantly increases the possibility of network blocking.

To reduce network blocking, researchers have turned to the traffic-aware (or congestion-aware) routing algorithms [6, 7, 8, 9, 10, 11], which make routing decisions based on network status. Generally, most of them are based on fully adaptive routing algorithms, so there is always possibility of making selection. However, existing fully adaptive routing algorithms may require a large number of virtual channels (VC) [12, 13, 14], or assume a conservative flow control technique [15, 16, 17].

The VC is expensive in the NoC scenario, although it was viewed as cheap and abundant [18]. First, adding VCs often means increasing router latency [19], because VA (VC allocation) delay increases with the number of VCs [20] and at most times VA is the critical stage of virtual-channel routers [19]. Secondly, adding VCs often means increasing router

[1]http://www.tilera.com/about_tilera/press-releases/tilera-double-processor-cores-every-two-years

area, because buffers are the main contributor to router area and adding VCs often requires more buffers. Algorithms based on Duato's theory [15] assume a conservative flow control technique that a queue never contains flits belonging to different packets [15, 16, 17]. This helps to reduce the number of VCs, but usually degrades the performance of networks carrying many back-to-back short packets [19], e.g., the control packets for cache coherence.

Compared with fully adaptive routing algorithms, partially adaptive routing algorithms without VCs are more cost-efficient and can be realized with aggressive flow control techniques [21, 22]. Glass and Ni [21] presented the turn model for designing partially adaptive routing algorithms without VCs. The turn model classifies all possible turns into clockwise and counter-clockwise abstract cycles, and prohibits one turn in each cycle to avoid deadlock. Based on the turn model, three partially adaptive routing algorithms: west-first, north-last, and negative-first, were proposed. Unfortunately, the degree of adaptiveness provided by above algorithms is highly uneven. To tackle this problem, Chiu [22] proposed the odd-even turn model, where EN [2] and ES turns are prohibited in even columns [3], and NW and SW turns are prohibited in odd columns. Odd-even routing could provide more even adaptiveness, but at the price of that none of long-distance (>2 hops) node pairs are provided with full adaptiveness.

Both uneven and insufficient routing adaptiveness, of state-of-the-art partially adaptive routing algorithms, may degrade the network performance. Applications' traffic tends to be very bursty [5], and the location of hot spots moves as time goes by. To avoid congestions, we expect the packets towards the hot spots are provided with full adaptiveness. Furthermore, since hot spots vary all the time, routing algorithm is expected to be able to provide full adaptiveness to all node pairs. Thus, none of the state-of-the-art partially adaptive routing algorithms could meet applications' requirements.

Based on above observations, the expected routing algorithm should be:

1. time/space-efficient, i.e., no VC and routing table requirement,
2. able to provide full adaptiveness to all node pairs.

It has been proved that it is impossible to design a deadlock-free fully adaptive routing without VCs for a mesh network [21]. Thus, the expected routing should be partially adaptive. Partially adaptive routing algorithms cannot provide full adaptiveness to all node pairs at all times, so that the second requirement is reduced to "able to" not "always" provide full adaptiveness to all node pairs. In other words, the routing algorithms could provide full adaptiveness to some node pairs at some time, and to other pairs when the traffic pattern changes.

To achieve such objectives, the topology[23] or routing should be reconfigurable [4]. Most of the state-of-the-art reconfigurable routing algorithms are designed for fault-tolerance, such as [27, 28, 29]. The common feature of them is that the reconfiguration is triggered by the detection of faults. If we simply declare the nodes belonging to congestion regions as faulty (temporally), the packets could be routed without entering into the congestion regions. However, according to fault-tolerant routing algorithms [27, 28, 29], faulty nodes, actually those in congestion regions, are not allowed to send and receive packets, otherwise the network maybe deadlock. As we analyzed above, hot-spot nodes are prone to be included into congestion regions, but we cannot forbid sending packets to them.

Above observations strongly encourage us to design a new reconfigurable routing algorithm, but this is a challenging job due to the deadlock problem. When the hot-spot nodes are detected, there is no existing principle to follow to generate the new deadlock-free routing algorithm. In general, deadlock happens when packets wait for each other in a cycle. To prevent deadlock, the routing algorithms without VCs should keep the channel dependence graph (CDG) acyclic [30]. So far, most of the cycle elimination algorithms, such as [31, 32, 33, 34], are off-line solutions. On-line dynamically eliminating CDG cycles is almost impossible. Turn models [21, 22] provide an elegant way to keep the CDG acyclic, but they are all static and cannot be directly used to generate reconfigurable routing.

To tackle this challenge, we inherit the core idea of turn models that keeping network deadlock free by breaking abstract cycles, and propose a dynamic turn model, namely the Abacus Turn Model (AbTM). AbTM compares a 2D mesh network to an abacus [5]. Imaging each column of a mesh is a wire with two sliding *beads*: clockwise *bead* and counter-clockwise *bead*. Actually, the *beads* represent the crunch points. Routers above and below *beads* are required to prohibit different turns in a way that CDG cycles cannot be formed. Thus, wherever the *beads* are located, the network is deadlock free. When the *beads* move along the column, the turns allowed and prohibited by routers vary as the relative position changes. In fact, turns are resources and reflect the ability of a router to forward packets, i.e., the adaptiveness. If a turn is enabled after the *beads* movement, corresponding routing adaptiveness is increased. Otherwise, the adaptiveness is reduced. Therefore, when the traffic pattern changes, we could adjust routing adaptiveness by moving *beads* inside each column. Furthermore, a safe operation is proposed to guarantee each step of *bead* movement correct, and two reconfigurable routing algorithms based on the AbTM are presented.

The rest of this paper is organized as follows: Section 2 discusses the proposed AbTM and the related safe operation, *bead passing*; Section 3 discusses the proposed two reconfigurable routing algorithms; Section 4 proves the claimed characteristic of AbTM-based routing algorithms; Section 5 evaluate the proposed solution; Section 6 discusses possible

[2] An EN turn is made when a packet changes its direction from east to north [21].

[3] A column is called even (respectively, odd) column if its coordinate in dimension-x is an even (respectively, odd) number [22].

[4] We should distinguish the reconfigurable routing algorithms from the routing reconfiguration algorithms. The former, reconfigurable routing algorithm, indicates that the routing algorithm could adjust itself to adapt to network variations, such as the detection of faults or hot-spot nodes. On the other hand, the routing reconfiguration algorithms are proposed to statically [24, 25] or dynamically [26] load a new routing algorithm, which was usually computed offline, to replace the old one without introducing deadlock.

[5] Although the proposed AbTM is topology agnostic, we limit our discussion on meshes for simplicity. At the end of this paper, we will show the way to implement AbTM on other topologies.

extensions; Section 7 distinguishes the proposed technique from related work; finally, Section 8 concludes this paper.

2. ABACUS TURN MODEL

In this section, we will discuss the rules defined by AbTM and show how do the AbTM-based reconfigurable algorithms reconfigure themselves through an illustrative example. Note that we interchangeably utilize terms, node and router, in the following of this paper.

2.1 The Basic Idea

According to the turn model proposed by Glass and Ni [21], a network is deadlock free if turns do not form any abstract cycle. Chiu [22] extended the turn model, and proved that a network is deadlock free if the rightmost columns in both clockwise and counter-clockwise abstract cycles never appear. As shown in Figure 1(a), the clockwise rightmost column consists of two allowed turns, ES and SW, connected by several contiguous NS channels from north to south. To remove the clockwise rightmost column, Chiu [22] requires the routers in odd columns to disable SW turn as shown in Figure 1(b), and those in even columns to disable ES turn as shown in Figure 1(c). The counter-clockwise rightmost column is removed by forbidding NW turn in odd columns and EN turn in even columns.

However, the odd-even turn model [22] is static, i.e., nodes cannot change the types of allowed and prohibited turns at runtime. In this paper, we inherit the basic idea of [22] that "a network is deadlock free if both clockwise and counter-clockwise rightmost columns are removed from the network", but we propose a more flexible way to realize it. As shown in Figure 1(a), to form a clockwise rightmost column, the ES turn should be above an SW turn. Hence, we could remove the clockwise rightmost column by forbidding all ES turns above any SW turn as shown in Figure 1(d). Therefore, in each column, there is a node, above which all ES turns are prohibited, and below which all SW turns are prohibited. In this way, no ES turn is above SW turn in any column, thus no clockwise rightmost column can be formed. We name this kind of node as clockwise *bead*. Similarly, in each column, there is also a counter-clockwise *bead*, above which NW turn is forbidden and below which EN turn is forbidden. Thus, the counter-clockwise rightmost column is removed.

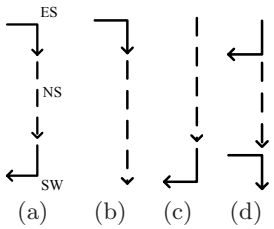
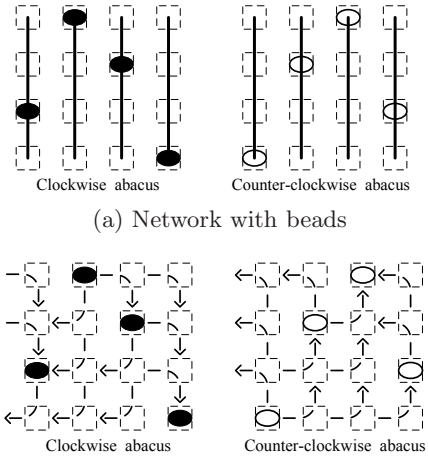


Figure 1: Clockwise rightmost column. (a) ES above SW. (b) ES only. (c) SW only. (d) ES below SW.

2.2 The Abacus Turn Model

As shown in Figure 2(a), a 4×4 mesh is compared to an abacus. Each column is viewed as a wire with two sliding *beads*: a clockwise *bead* and a counter-clockwise *bead*. Clockwise (resp., counter-clockwise) *beads*, which are separately controlled, are utilized to regulate the distribution of clockwise turns (resp., counter-clockwise turns). In general, AbTM rules are shown in the following:

1. nodes above clockwise (resp., counter-clockwise) bead forbid ES (resp., NW) turn,
2. nodes below clockwise (resp., counter-clockwise) bead forbid SW (resp., EN) turn,
3. clockwise (resp., counter-clockwise) bead holer does not prohibit clockwise (resp., counter-clockwise) turns.



(a) Network with beads

(b) Network marked with prohibited turns

Figure 2: An AbTM example. Dashed blocks represent possible position for beads, solid ellipses represent clockwise beads, hollow ellipses represent counter-clockwise beads, dashed arrows indicate the prohibited turns, and allowed turns are not shown for simplicity.

Figure 2(b) shows the distribution of prohibited turns that are represented by dashed arrows. According to the AbTM, clockwise (resp., counter-clockwise) bead holder allows all clockwise turns: ES, SW, WN, and NE (resp., all counter-clockwise turns: EN, NW, WS, and SE). The router above clockwise (resp., counter-clockwise) bead allows all clockwise (resp., counter-clockwise) turns except the ES (resp., NW) turn. On the other hand, the router below clockwise (resp., counter-clockwise) bead allows all clockwise (resp., counter-clockwise) turns except the SW (resp., EN) turn. Turns are distributed in such a way that both clockwise and counter-clockwise rightmost columns cannot be formed. Based on AbTM, designing a deadlock-free routing algorithm is reduced to assigning the positions of clockwise and counter-clockwise *beads* inside each column. Once the positions are fixed, the network is deadlock-free. We prove this as the following theorem.

Theorem 1. *The network following abacus turn model is deadlock free.*

PROOF. We prove this theorem by contradiction. We assume a network following *abacus* turn model enters into deadlock, so that there is a turn dependence cycle in the network according to [21]. Moreover, according to [22], there must be a clockwise or counter-clockwise rightmost column in this cycle. Without loss of generality, we assume it is

a clockwise rightmost column. Therefore, there must be a node, N_a, allowing the ES turn is above a node, N_b, allowing the SW turn. According to the *abacus* turn model, N_a should be the clockwise *bead* or below it, and N_b should be the clockwise *bead* or above it. Therefore, N_a cannot be above N_b. Contradiction arises. □

For each $k \times k$ mesh, there are two beads in each column and k potential positions for each *bead*, thus there are $k \times k$ combinations for each column and $(k \times k)^k$ combinations for a network. Each combination of *beads* represents a routing configuration, so that routing reconfiguration is reduced to moving up/down the *beads* inside each column.

Example As shown in Figure 3(a), clockwise *beads* are located on the bottom row initially. Thus, routers above *beads* prohibit the ES turn. Meanwhile, a new hot spot, node-5, is detected, and node-6 is expected to send packets to it in a relatively long period. Since there is only one available path between them [6], this path is highly prone to congestions. Node-6 wants to have more available paths to balance traffic, so it makes a complaint to node-7 about the prohibited ES turn at each time it tries to send packets. Node-7 collects the complaints, and negotiates with the bead holder, node-1. The holder will evaluate the requirements, and determine whether to give up the *bead*. In this example, node-1 will pass the *bead* up as shown in Figure 3(b). Thus, node-7 could enable the ES turn according to AbTM, and node-6 has two available paths for traffic balancing. In the same way, node-7 will make complaints to node-8. Finally, node-8 also gets the ownership of the clockwise *bead* as shown in Figure 3(c). By now, packets from node-6 to node-5 could utilize all minimal paths to avoid congestion.

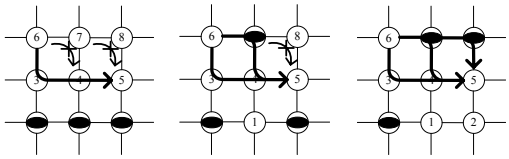


Figure 3: AbTM reconfigurable routing example.

2.3 Bead Passing

Moving *beads* is also a hard job due to the difficulty of keeping the network deadlock free. For example, as shown in Figure 4(a), node-1 holds the clockwise *bead*. Thus, it allows the SW turn, and node-4 could utilize it to forward packets to node-0. If the *bead* is pulled up, as shown in Figure 4(b), node-1 will prohibit SW turn. If node-4 does not notice that change in time, the packets sent to node-0 could be blocked at node-1. Furthermore, if node-7 enables ES turn when the SW turns, to be disabled by node-1 and node-4, are still hold by packets, a clockwise rightmost column maybe formed and the AbTM is violated.

To address above problems, *bead* movement should follow two rules:

1. a turn cannot be disabled unless no packet will require it,

[6]We assume the reconfigurable routing algorithm is minimal. At the end of this paper, we will show the way to implement non-minimal routing based on AbTM.

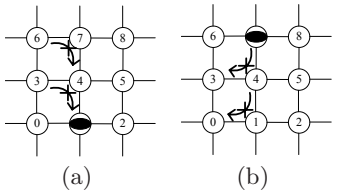


Figure 4: Prohibited turns distribution, (a) before bead movement, (b) after bead movement.

2. a turn cannot be enabled unless no packet is holding the other turn needed to form the rightmost column.

Generally, to move a *bead* h hops, h turns should be disabled and h other turns should be enabled. For example, to move *bead* from node-1 to node-7, the SW turns on node-1 and node-4 should be disabled, and the ES turns on node-4 and node-7 should be enabled. Thus, it is a hard job to meet above two rules as a large number of turns will be involved, especially in a large-scale network. To reduce the complexity of *bead* movement, we divide it into h steps, within which the *bead* is moved up/down just one hop. This basic step is viewed as a safe operation and is named as *bead passing*. Exploiting this safe operation has two advantages. First, *bead passing* can be locally realized by the cooperation of adjacent routers leading to a good scalability. Second, *bead passing* itself can guarantee the network deadlock free during the reconfiguration, so that designers do not need to consider the deadlock problem when designing their own reconfigurable routing algorithms.

Within each step, only one turn should be disabled at original *bead* holder. For example, to move *bead* from node-1 to node-4, node-1's SW turn should be disabled. To do that, node-1 should first notify node-4 to stop injecting southwest packets [7] because they may require the SW turn on node-1. When node-4 receives the notification, it labels its south output port as "southwest unaccepted". After then, all southwest packets will be routed to node-3, instead of node-1, because node-3 always allows the WS turn. However, currently, node-4 cannot send acknowledgement to node-1, because there maybe southwest packets have already been routed based on the old information. Thus, node-4 should drain such kind of packets first, and then send acknowledgement to node-1.

Draining packets is an extensively-studied topic in the field of reconfiguration algorithms [24, 25, 26]. In this paper, we borrow the idea of [25] that exploits the reconfiguration token. The difference is that not all packets are required to be drained. In above example, node-4 only has to drain southwest packets. Since node-4 only receives southwest packets from its local, east, and north input ports, reconfiguration tokens can only be inserted into the end of the queues belonging to these three input ports. When the south output port receives all these three tokens, node-4 can conclude that there is no southwest packets routed to node-1 existed in itself. Thus, it could send acknowledgement to node-1 at that time.

When node-1 receives the acknowledgement from node-4, it has the confidence that no new southwest packet will be received from its north neighbor. However, there maybe southwest packets still queued in the north input port. Thus,

[7]Southwest packets are those whose destination is on the southwest to current node.

it should first drain the southwest packets in itself before disabling SW turn. After then, it could disable the SW turn and pass the *bead* up. when node-4 receives the *bead*, it enables its ES turn and notifies its west neighbor, node-3, that it could receive southeast packets from now on. By now, one iteration of *bead passing* finishes. Above operations are summarized in the line-2 to line-5 in Figure 5(a). Moving down the clockwise *beads* and moving up/down counter-clockwise *beads* follow the same procedure, but different packets should be drained in different neighbors as shown in Figure 5.

Theorem 2. *bead passing does not introduce deadlock into network.*

PROOF. We prove this theorem by contradiction. To distinguish packets routed according to old and updated routing algorithms, we refer them as old packets and new packets respectively. We assume a network enters into deadlock during *bead passing*. According to [22], there is a clockwise or counter-clockwise rightmost column. Without loss of generality, we assume it is a clockwise rightmost column. Thus, there should be a packet, P_a, taking an ES turn is above another packet, P_b, taking an SW turn. There are four different kinds of combinations of P_a and P_b: {new, new}, {new, old}, {old, new}, and {old, old}. Because the network following the AbTM, so that the combination cannot be {new, new} and {old, old} as we proved above. If the combination is {new, old}, P_a must take this ES turn on the new clockwise *bead* because nodes above it do not allow the ES turn, P_b must take this SW turn on the old *bead* because nodes below it do not allow the SW turn, and the *bead* is moving up. Therefore, P_a takes ES turn **after** the declaration of new *bead* and P_b takes SW turn **before** the declaration. Hence, combination {new, old} is impossible. Due to the similar reason, combination {old, new} is impossible neither. Contradiction arises. □

3. THE RECONFIGURABLE ROUTING ALGORITHMS

Based on AbTM and the safe operation, designing a reconfigurable routing algorithm is reduced to defining a way to determine the direction of *bead* movement. In this section, we will discuss two possible algorithms. Furthermore, implementation issues of reconfigurable routing algorithms are discussed at the end of this section.

3.1 Arm-wrestling

According to the AbTM, four turns, WN, NE, WS, and SE, are not critical and always assumed to be enabled. Other four turns, ES, SW, EN, and NW, are updated by *bead passing*. These four turns can be classified into two groups, clockwise group, ES and SW, and counter-clockwise group, EN and NW. Basically, moving *bead* means disabling a turn in the old *bead* holder and enabling the other turn in the same group in the new holder. Therefore, the turn requirement is a nature metric to determine the direction of *bead* movement.

With the requirements of each turn, moving *bead* can be compared to moving a block (mass is 0) pressed onto a wall as shown in Figure 6. The *bead* is compared to the block. To pull up the clockwise *bead*, for example, the north neighbor give a force, F_{up}, which reflects its eagerness for ES turn. On the other hand, the south neighbor give a force, F_{down}, which reflects its eagerness for SW turn. The current holder gives a friction, f, to prevent the motion. The direction of f is always opposite to the direction of *bead* movement. If $F_{up} > F_{down}$, the *bead* is going to be moved up. Thus, to prevent that movement, f reflects holder's eagerness for SW turn. Otherwise, it reflects its eagerness for ES turn to prevent moving down the *bead*. Furthermore, to prevent "shaking" that a *bead* is frequently moved up and down, we set a threshold (Th). Therefore, to move the *bead*, the force summation should be bigger than the Th. For moving counter-clockwise *beads*, F_{up} will reflect north neighbor's eagerness for NW turn, F_{down} reflects south neighbor's eagerness for EN turn.

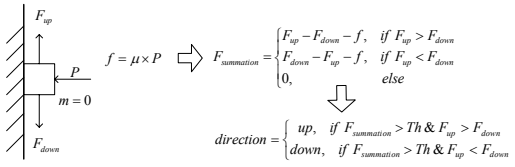


Figure 6: An example of moving up/down a block.

3.2 Tug-war

With the *arm-wrestling* algorithm, beside the requirement of current holder, only the direct north and south neighbors are taken into consideration. Therefore, the requirements of farther neighbors maybe not fulfilled in time. To address this problem, we propose the *tug-war* algorithm that differs with *Arm-wrestling* in the definition of F_{up} and F_{down}. With *tug-war* algorithm, the nodes above *bead* are viewed as a group, their total requirement to a corresponding turn is defined as the F_{up}. On the other hand, nodes below *bead* are viewed as an another group, and their total requirement to the turn is defined as the F_{down}. To get the total requirement, each router increases its local requirement by the requirement received from upstream nodes, and propagates the result to the downstream node through out-of-band dedicated wires. Furthermore, to highlight the requirement of nodes closer to *bead*, the total requirement is divided by 2 at each node.

3.3 Implementation Issues

To calculate the forces, e.g., F_{up} and F_{down}, each node remembers how many packets required corresponding turns in itself and how many "complaints" were received from neighbors. Particularly, a "complaint" is received when a forbidden turn is required by a neighbor.

Generally, there are two ways to implement routing algorithms, logic-based and table-based. Logic-based routing algorithms, such as xy routing, are usually time/space efficient. However, they are lack of flexibility for reconfiguration. Table-based solution is flexible, but it may introduce large area and timing overhead. Recently, a logic-based distributed routing (LBDR) is proposed to implement turn-based routing algorithms by utilizing 8 routing bits (R_{en}, R_{es}, R_{se}, R_{sw}, R_{wn}, R_{ws}, R_{ne}, and R_{nw}) and 4 connectivity bits (C_e, C_s, C_w, and C_n) [18]. The routing bits indicate whether the corresponding neighbor accepts such kind of packets. For example, $R_{en} = 1$ means that the east neighbor accepts the northeast packets (allows the EN turn). The connectivity bits represent whether the neighbor is connected. LBDR routing is separated into two steps. The first step

```
1. IF MoveUp
2.    Notify north neighbor to drain
         southwest packets in local,
         east, and north input queues.
3.    Wait for acknowledgement.
4.    Drain southwest packets in north
         input queue.
5.    Disable SW and Pass bead up
6. ELSIF MoveDown
7.    Notify west neighbor to drain
         southeast packets in local,
         west, and north input queues.
8.    Wait for acknowledgement.
9.    Drain southeast packets in west
         input queue.
10.   Disable ES and Pass bead down
11. ENDIF
```

(a) clockwise bead passing

```
1. IF MoveUp
2.    Notify west neighbor to drain
         northeast packets in local,
         west, and south input queues.
3.    Wait for acknowledgement.
4.    Drain northeast packets in west
         input queue.
5.    Disable EN and Pass bead up
6. ELSIF MoveDown
7.    Notify south neighbor to drain
         northwest packets in local,
         east, and south input queues.
8.    Wait for acknowledgement.
9.    Drain northwest packets in south
         input queue.
10.   Disable NW and Pass bead down
11. ENDIF
```

(b) counter-clockwise bead passing

Figure 5: Pseudo code of bead passing.

compares the coordinates of current router and destination router. The results point out the directions which can take the packet closer to the destination. The second step checks the routing and connectivity bits, and determines whether the corresponding output ports can be taken. For example, the east output can be taken if the east neighbor is connected and one of the following three conditions holds

1. the destination is on the east of the current router,
2. the destination is on the northeast and the east neighbor allows the EN turn,
3. the destination is on the southeast and the east neighbor allows the ES turn.

LBDR routing provides just enough flexibility to implement AbTM-based reconfigurable routing algorithms. Since AbTM assumes WN, NE, ES, and SE turns are always enabled, we could remove 4 routing bits, R_{se}, R_{wn}, R_{ws}, and R_{ne}. Furthermore, a reconfiguration module, which realizes the *bead-passing*-based *arm-wrestling* or *tug-war* algorithms, should be added. The reconfiguration module is utilized to update these four routing bits based on the runtime network status according to *arm-wrestling* or *tug-war* algorithms. Since reconfiguration module does not add operation to the critical path of LBDR routing, the time efficiency of LBDR routing is inherited.

4. PROOFS

In this section, we prove the claimed characteristics of AbTM-based routing, i.e., deadlock-free, livelock-free, and no packets will be dropped or suspended during the reconfiguration. Before the discussion, there is an important assumption that the network should be initialized to be deadlock-free and LBDR-connected. LBDR-connected means the network topology is supported by LBDR, and initially there is at least one minimal LBDR-path existed for any two nodes.

LEMMA 1. *AbTM-based routing always provide at least one minimal path for any two nodes at any time*

PROOF. According to AbTM, the WN, NE, WS, and SE turns are always enabled at each node. Therefore, for any two nodes, if AbTM-based routing could provide a minimal path consisted of these four turns only, the lemma is proved. Otherwise, we should prove that the critical turns (ES, SW, EN, and NW) in the path will never be disabled.

First, we assume the topology is a regular mesh, as shown in Figure 7(a). In this case, for any node, such as node-4, any other node can be reached utilizing uncritical turns. For example, to reach northeast destination, such as node-8, AbTM-based routing could route the packet to north until the intermediate node is on the same row as the destination, and then route it to the destination by taking a NE turn. For other directions, the critical turns are not required neither. Therefore, there is always a minimal path between any two nodes at any time.

Second, we assume there are two nodes are not connected unless utilizing a critical turn. Without loss of generality, we assume the critical turn is ES turn. In this case, there must be an intermediate node without south neighbor ($C_s = 0$), such as the node-4 shown in Figure 7(b). Otherwise, this intermediate node could send the packet to its south neighbor because SE turn is always enabled. Since any two nodes are initially connected, the ES turn at node-5 should be enabled. Therefore, the clockwise *bead* is node-5 or above it. Without loss of generality, we assume node-5 holds clockwise *bead*. To disable the ES turn at node-5, the *bead* should be pulled down. According to the description of *arm-wrestling* and *tug-war*, the SW turn requirement of nodes below node-5 should be larger than the ES turn requirement of nodes above node-5. However, the SW requirements of nodes below node-5 is always 0 because they do not have west neighbor. Therefore, clockwise *bead* will never be below node-5, then the ES turn at node-5 is always enabled.

The proofs for node pairs requiring SW, EN, and NW turns are similar with above proof for ES turn, and are omitted due to the limited space. We should note that, in Figure 7(c), node-4 does not have the west neighbor, so that the SW turn at node-1 cannot be disabled. In this case, the ES turn requirements of node-7 is also 0 because it does not have west neighbor neither. Otherwise, if the node-6 exits, the network topology is a horizontally-reversed "C" that is not supported by LBDR routing. In Figure 7(d), node-7 does not have the west neighbor due to the same reason. □

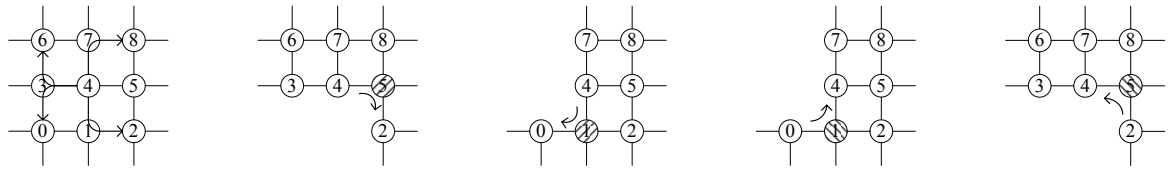


Figure 7: Nodes are always connected in AbTM-based routing.

Theorem 3. *AbTM-based routing does not drop and suspend packets during the reconfiguration.*

PROOF. In general, a packet is dropped at an intermediate node if there is no route for that packet. According to Lemma 1, AbTM-based routing always provides at least one minimal path for any node pair. Therefore, dropping packet will not happen.

According to Lemma 1 there is always a minimal path between any two nodes. This path will not be changed by the reconfiguration. Therefore, for any packet, it does not need to wait the reconfiguration to finish, and can proceed along the unchanged minimal path. Hence, no packet will be suspended during the reconfiguration. □

Theorem 4. *AbTM-based routing is deadlock and livelock free.*

PROOF. Deadlock-freeness can be directly proved by Theorem 1 and Theorem 2.

AbTM-based routing is a minimal routing, so that a packet will get closer to its destination at each hop. Furthermore, Theorem 3 shows that no packet is suspended, thus all packets will definitely reach their destinations. □

5. EVALUATION

In this section, we will evaluate AbTM-based reconfigurable routing algorithms, such as the proposed *arm-wrestling* and *tug-war*, by comparing them with other state-of-the-art routing algorithms. Specifically, network performance and implementation overhead are evaluated in this section.

5.1 Methodology

AbTM provides a safe way to dynamically tune the routing algorithm, where "safe" means that the reconfiguration and reconfigured routing algorithm are deadlock free. Thus, we select four baseline routing algorithms which were also proposed to address the "safe" problem. These algorithms include one deterministic routing algorithm: xy routing, two partially adaptive routing algorithms: west-first [21] and odd-even routing [22], and a minimal fully adaptive routing [15]. Recently proposed routing algorithms, such as C-QR [35], O1Turn [36], and RCA [10], are proposed to improve the load balancing (or address the "port selection" problem). Thus, AbTM is orthogonal to them.

Adaptive routing algorithms may produce two candidate output ports. In such cases, the output port with more credits is selected. Each router is assumed to have 5 input and output ports. All routers, except [15], assume one VC per virtual network, and each VC is assigned with a 4-entries queue. For [15], two VCs per virtual network are provided to avoid deadlock. For fair comparison, its input buffer queue is equipped with 2 entries. Furthermore, [15] does not allow an VC to be reallocated until the tail flit leaves. For other routing algorithms, however, an VC can be reallocated once the tail flit is stored.

We first evaluate the routing algorithms under synthetic traffic patterns, including uniform, transpose and hotspot, in a 4×4 mesh. After then, we simulate them again in a 8×8 mesh to show their scalability. All of these algorithms are implemented within the cycle accurate Garnet simulator [37]. Garnet provides two simulation modes, flexible and detailed. In our simulation, we adopt the detailed mode, because it enables us to modify the router architecture for reconfiguration. Routers realize one virtual network per physical channel in this simulation.

We also evaluate these routing algorithms under real applications's traffic patterns through a trace-driven simulation. We use the GEMS simulator [38], which is based on a full-system functional simulation infrastructure, i.e., the Simics [39], to trace applications' traffic patterns. Simics could run unmodified Solaries operating system and guarantee the function correctness. GEMS, on the other hand, provides the timing modules of memory system and microprocessors, i.e., the Ruby and Opal modules. In our simulations, we only utilize the Ruby module, which includes the Garnet simulator [37]. The cache coherence protocol, MSI_MOSI_CMP_directory, is synthesized. This protocol requires five virtual networks to avoid deadlock, and two of them are ordered [8]. The network is assumed as 4×4 mesh, instead of the 8×8 mesh, to save simulation time. Benchmark suites, Splash-2 [41] and PARSEC [42], are evaluated in this simulation. As for the threshold (Th) used to prevent bead shaking, we set it as 20 in this simulation as averagely about 10 cycles will be cost to finish one *bead passing* according to our simulation. We manually set the value of Th to show the potential of the reconfigurable routing. We think finding a theoretical or dynamical way to determine the best Th is an interesting and challenge future work.

Finally, to reveal the cost of AbTM-based routing algorithms for the performance gain, we analyze its timing, area and power overhead in the end of this section. In this simulation, we implement six different routers realizing xy, west-first [21], odd-even [22], minimal fully adaptive routing [15], and AbTM-based routing algorithms respectively. We utilize the Design Compiler to synthesize the verilog codes with the UMC 90nm standard cell library. Each router, except [15], consists of five input/output ports, each input port has one VC, which is provided with a 4-entries queue. For [15], each input port has 2 VCs with a 2-entries queue. The width of input buffer queue is assumed as 64 bits, both VC and switch allocators adopt the round-robin arbiter proposed in [20], and an fully connected 5×5 crossbar is implemented.

[8]Techniques have been proposed to keep packets' ordering in adaptive routing [40]

5.2 Simulation Results and Analysis

5.2.1 Performance Under Synthetic Traffic Patterns

Under the uniform traffic pattern, each node sends packets to others with the same possibility. The simulation results for 4×4 and 8×8 meshes are shown in Figure 8(a) and Figure 8(b), respectively. Horizontal axis represents the flit injection rate, and the vertical axis indicates the average packet latency in router cycles. It has been widely proved that deterministic routing algorithms could get better performance than adaptive routing algorithms under the uniform traffic pattern [21, 22]. The main reason for that phenomenon is that adaptive routing algorithms usually make selection based on local information, such as the number of credit of each output port. This kind of short-sight solution usually degrade network performance. According to the results shown in Figure 8(a), xy routing algorithm gets the best results as expected. Two partially adaptive routing algorithms, west-first and odd-even, get similar results. The results of AbTM-based reconfigurable routing algorithms are worse than that of partially adaptive routing algorithms mainly due to the reconfiguration strategy. Because AbTM-based reconfigurable routing algorithms reconfigure themselves mainly based on the traffic history. However, under uniform traffic pattern, such kind of reconfiguration will be wrong at most times. For example, we assume northeast direction carries the highest traffic load in current interval, so that AbTM-based will allocate more adaptivity to that direction. However, according to the definition of uniform traffic, northeast direction will carry the lowest traffic load in the next interval with very high possibility. The minimal fully adaptive routing algorithm [15], labeled as "min-adapt", gets the worst results due to its conservative flow control strategy that an VC cannot be reallocated until it is empty. We also simulate these algorithms under a 8×8 mesh, and the results are shown in Figure 8(b). Most of the results agree with that shown in Figure 8(a). However, the relative performance of min-adapt routing algorithm get improved in this simulation. The reason is that the contention possibility increases with the network size. When contention happens, [15] requires packets to wait on the escape output port, i.e., the one chosen by xy routing. This characteristic enable min-adapt routing algorithm to exploit the long-term evenness of uniform traffic pattern just like the xy routing.

In real world, applications usually generate nonuniform traffic patterns, such as transpose. Under transpose traffic pattern, source node s always sends packets to destination d, where $d_i = s_{(i+b/2)\%b}$ and b is the number of bits used to index nodes. According to the simulation results shown in Figure 8(c), the AbTM-based routing algorithms get the best results because they could provide full adaptiveness to packets. Transpose traffic pattern tends to cause severe congestion problems, thus the xy routing gets the worst performance. West-first routing gets better results than xy routing as it could provide full adaptiveness to eastward packets. However, the improvement is limited because westward packets still face the congestion problem. Odd-even routing could provide even adaptiveness to packets towards different directions, thus it could get better results than west-first and xy routing. However, the provided adaptiveness is insufficient, thus it cannot get the performance as high as the AbTM-based routing algorithms do. Min-adapt routing algorithm also could provide full adaptiveness, but its conservative flow control technique will aggregate the congestion problem, thus it saturates earlier than the proposed AbTM-based routing algorithms. We also simulate these algorithms in a 8×8 mesh, the results are shown in Figure 8(d). Relative performance does not change, but the improvement got by the proposed algorithms is enlarged. Because its provided full adaptiveness is useful to tackle the more severer congestion problem in the large network.

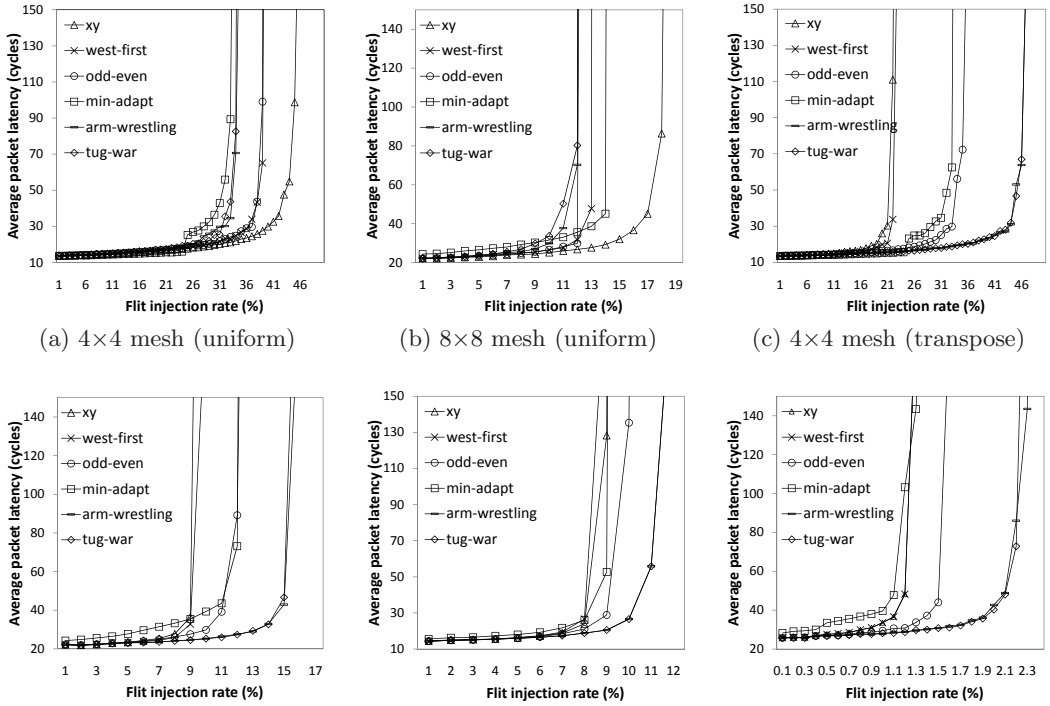
Applications' bursty traffic may cause hotspots, and aggregates the congestion problem. In the following two simulations, we assume four hot-spot nodes: n_0, n_4, n_8, n_{12}, each of which gets extra 20% traffic than others. These four nodes are selected to simulate the situation that four memory controllers are frequently accessed. In such cases, westward packets are prone to be congested. According to simulation results shown in Figure 8(e), the AbTM-based routing algorithms get the best results because they could provide full adaptiveness by enabling the NW and SW turns at every routers. Although min-adapt routing algorithm could provide full adaptiveness, but its conservative flow control technique leads to a low utilization of input buffers. Thus, its performance is not better than odd-even routing algorithm. Since west-first and xy routing algorithms cannot provide adaptiveness for avoiding congestions, they get the worst performance. When the network is enlarged, the congestion problem is further aggregated. By successfully reducing blocking, the AbTM-based routing algorithms also get the performance. Furthermore, the improvement, compared with other algorithms, is also enlarged as they did under transpose traffic pattern.

5.2.2 Performance Under Applications' Traffic

According to above simulations, we find that *arm-wrestling* and *tug-war* routing algorithms could get good performance under nonuniform traffic patterns, such as transpose and hotspot. In a short range perspective, there are lots of bursts in application's traffic [43]. If we view burst as hotspot traffic, then application is consisted of hotspot traffic in consecutive intervals with different hot-spot nodes. If AbTM-based routing could reconfigure itself in time, we could expect the application performance improvement. To prove this assumption, we further carried out the following trace-driven simulation.

Figure 9(a) shows the packet latency across Splash-2 benchmarks normalized to the latency of xy routing. By dynamically allocating more adaptivity to bursty packets, the AbTM-based routing could significantly reduce the packet latency. In general, *arm-wrestling* and *tug-war* routing algorithms could reduce packet latency for all applications. However, due to the different characteristics of applications, the improvement is different. For example, for applications tend to generate high contented traffic, such as fft and water-spatial, the improvement is significant. However, for applications that generate low contented traffic, such as raytrace and ocean, the improvement will be relative small. For example, for the Ocean applications, where more than 60% traffic is local traffic [41], *arm-wrestling* algorithm could reduce the packet latency by 6% and *tug-war* algorithm reduces it by 7%. In general, for Splash-2 applications, AbTM-based routing algorithms could reduce the packet latency by maximally 19% and 10% in average.

To improve the confidence to make the conclusion that AbTM-based routing algorithms outperform other algorithm-



(a) 4×4 mesh (uniform) (b) 8×8 mesh (uniform) (c) 4×4 mesh (transpose)

(d) 8×8 mesh (transpose) (e) 4×4 mesh (hotspot) (f) 8×8 mesh (hotspot)

Figure 8: Average packet latency under synthetic traffic patterns.

s, we simulate them again under PARSEC benchmarks [42]. The simulation results, which are also normalized to the latency of xy routing, are shown in Figure 9(b). Simulation results point to the same conclusion that AbTM-based routing algorithms could significantly improve the network performance. Specifically, for high-contented applications, such as canneal and freqmine, the improvement is large. For example, for freqmine, *tug-war* reduces the packet latency by 15%, and *arm-wrestling* reduces it by 12%. However, for low-contented applications, the reduction is moderate. For example, for streamcluster, the packet latency reduction is 2% and 3% by *arm-wrestling* and *tug-war*, respectively. In general, the packet latency is reduced by 15% maximally and 10% in average.

5.2.3 Overhead

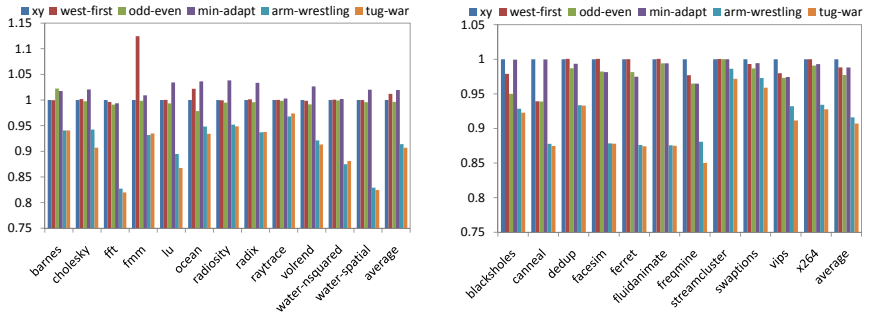
The router frequency is one of the most important performance metrics to evaluate NoCs, and in general, it should be kept as high as possible. As discussed in Section 3.3, AbTM-based routing algorithms provide a solution to dynamically update routing bits of LBDR routing, but does not modify its routing logic. This together with the fact that LBDR routing has been proved to be timing efficient [18] prove that the AbTM-based routing algorithms are also timing efficient.

Another kind of overhead is the area overhead, which determines the chip cost. As shown in Figure 10(a), the router area is normalized to the area of xy router. According to the experiment results, the router realizing min-adapt routing algorithm is the largest because it requires 5 VC allocators. Other routers do not implement virtual channels, so that they do not have the VC allocator. The router realizing *tug-war* routing algorithms is the second largest, because it requires more logics to propagate traffic information. *Arm-wrestling* router requires fewer logics than *tug-war*, but is still larger than west-first and odd-even router. Compared with xy router, *Arm-wrestling* router increases the area by 5% and *tug-war* increases it by 8%. Taking the area of core and caches into consideration, the area increase is negligible. As shown in Figure 10(b), if the router area is assumed to occupy 11% area of a tile as reported by Intel [1], then the increase of tile area will be smaller than 1%.

Assuming the same VLSI technology and working frequency, the dynamic power is largely determined by the size and activity of routers. As discussed above, the routers do not have significant difference in area. Thus, they consume similar dynamic power as the activity of them does not have significant difference either. Reconfiguration also consumes power, but it is rare to happen compared with routing operations. If we further take the power of cores and caches into consideration, the power overhead is negligible.

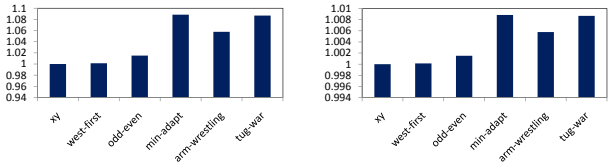
6. DISCUSSIONS

AbTM-based routing can be extended to any topology with the help of segment-based routing [31] which was initially proposed to design deadlock-free fault-tolerant routing algorithms. The segment-based routing divides a network into segments, and places bidirectional turn restrictions within each segment to keep the network deadlock-free. Currently, the segment-based routing requires off-line computation to determine how to divide the network and where to place the turn restrictions. With the AbTM-based routing, we require the processor allocation algorithms to provide the information about the segments. Thus, the AbTM-based routing could dynamically determines where to put the restrictions at runtime.



(a) Splash-2 benchmarks (b) PARSEC benchmarks

Figure 9: Packet latency across Splash-2 and PARSEC benchmarks.



(a) Router Area Overhead (b) Tile Area Overhead

Figure 10: Area overhead evaluation.

AbTM-based routing is minimal because the underlying LBDR is minimal. However, at some times, non-minimal routing is useful for balancing traffic and tolerating faults. Recently, Rodrigo et.al. have extended the LBDR to be non-minimal [44]. Since AbTM-based routing does not modify LBDR's routing logic, the extension proposed by Rodrigo et.al. can also be applied to AbTM-based routing.

The reconfiguration time, i.e., the cycles utilized to finish one reconfiguration, is usually evaluated to reflect the performance of reconfiguration algorithm. Compared with [25], since fewer packets are required to be drained, the reconfiguration of the proposed reconfigurable routing should be faster than [25]. Furthermore, because the proposed algorithm does not suspend packets, we did not evaluate the reconfiguration time in this paper due to the limited space.

7. RELATED WORK

7.1 Deadlock-Free Reconfiguration Algorithms

Generally, deadlock-free reconfiguration algorithms can be categorized into static algorithms [24, 25] and dynamic algorithms [26]. Traditional static algorithms avoid deadlock during the reconfiguration by stopping user traffic until the reconfiguration finishes [24]. As a consequence, significant network performance degradation is introduced. On the other hand, dynamic reconfiguration algorithms let user traffic continue while the network reconfigures (packets also should be suspended in regions affected by reconfiguration) [26]. However, they are either complex to implement or designed for specific routing algorithms, such as the up*/down* routing algorithm. Recently, an overlapping static reconfiguration algorithm is proposed to reduce the performance degradation caused by static reconfiguration while still maintain the simplicity [25]. Its basic idea is to globally overlap reconfiguration phases via locally synchronization. However, some packets still should be suspended until the local synchronization finishes.

The proposed reconfiguration algorithms are different with prior algorithms in the following aspects. First, most of the previous algorithms are proposed to handle the topology changes caused by router or link failures [24, 25, 26]. On the other hand, the proposed algorithms assume the network is fault free. This assumption is reasonable because allocation algorithms always schedule a fault-free subnetwork to incoming applications [45]. Second, previous algorithms should suspend packets, in the whole network [24] or regions affected by the reconfiguration [25, 26]. With the proposed algorithms, however, no packet will be suspended. Finally, no packet is dropped during the reconfiguration and all packets (old and new) will be correctly delivered to their destinations.

7.2 Application-/Traffic-/Congestion-Aware Routing Algorithms

Exploiting the traffic or application information is an effective way to improve routing performance [32, 33, 34, 14, 6, 7, 8, 9, 10, 11]. If the application's traffic information, such as the communication graph, can be obtained in advance, we could design a routing algorithm that not only keeps the network deadlock free but also maximally improve the network performance. This assumption usually holds in MPSoC scenarios, and the representative algorithms are the APSRA proposed by Palesi et.al. [32], the deadlock-free oblivious routing proposed by Kinsy et.al. [33], and the ACES proposed by Cong et.al. [34]. However, in the chip multiprocessor scenario, where the proposed AbTM-based routing focuses on, traffic/application information should be gathered at runtime. Existing routing algorithms exploiting runtime traffic information are designed for output port selection [14, 6, 7], congestion reduction [8, 9, 10], or pipeline reduction [11]. Thus, the proposed AbTM-based routing,

which is designed for on-line routing reconfiguration, is orthogonal to them.

8. CONCLUSION

Exploiting applications' traffic information to improve NoC performance has been proved to be effective. In this paper, we present an on-line routing reconfiguration strategy. It dynamically reconfigures itself to provide full adaptiveness to the highest traffic load directions. The major challenge addressed in this paper is the way to keep the network deadlock free during and after each reconfiguration. To this end, we propose the AbTM, which keeps the network, with a stable configuration, deadlock free. AbTM significantly reduces the complexity of routing reconfiguration. Furthermore, we propose an atomic reconfiguration operation, i.e., *bead passing*, which inherently keeps the network deadlock-free during the reconfiguration without suspending and dropping packets. Based on AbTM, two reconfigurable routing algorithms, *arm-wrestling* and *tug-war*, are proposed. The AbTM-based routing algorithms are proved to be deadlock- and livelock-free, and are evaluated against both synthetic and real application traffic. The experimental results are indeed encouraging and prove its applicability with scalable performance in large-scale NoC applications.

9. ACKNOWLEDGMENTS

The work was supported in part by National Basic Research Program of China (973) under grant No. 2011CB302503, in part by National Natural Science Foundation of China (NSFC) under grant No.(60806014, 61076037, 60921002), in part by Co-Building Program of Beijing Municipal Education Commission.

We would like to thank Prof. Kai Hwang of USC for his long time help, Prof. Ninghui Sun for his insight suggestions to our work, Dr. Kun Wang and Dr. Xiaotao Chang of the IBM CRL for their discussion, and the anonymous reviewers for their constructive comments.

10. REFERENCES

[1] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, 2008.

[2] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Chyi-Chang Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core soc with mesh interconnect. In *Digest of Technical Papers. IEEE International Solid-State Circuits Conference*, pages 88,89,598, 2008.

[3] W.J. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[4] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[5] N. Barrow-Williams, C. Fensch, and S. Moore. A communication characterisation of splash-2 and parsec. In *IEEE International Symposium on Workload Characterization*, pages 86–97, 2009.

[6] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, and C.R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of Design Automation Conference*, pages 559–564, 2005.

[7] Arjun Singh, W.J. Dally, A.K. Gupta, and B. Towles. Goal: a load-balanced adaptive routing algorithm for torus networks. In *Proceedings of International Symposium on Computer Architecture*, pages 194–205, 2003.

[8] J.W. van den Brand, C. Ciordas, K. Goossens, and T. Basten. Congestion-controlled best-effort communication for networks-on-chip. In *Design, Automation Test in Europe Conference Exhibition, 2007.*, pages 1–6, 2007.

[9] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks. In *Proceedings of International Symposium on High-Performance Computer Architecture*, pages 108–119, 2005.

[10] P. Gratz, B. Grot, and S.W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *Proceedings of IEEE International Symposium on High Performance Computer Architecture*, pages 203–214, 2008.

[11] D. Park, R. Das, C. Nicopoulos, Jongman Kim, N. Vijaykrishnan, R. Iyer, and C.R. Das. Design of a dynamic priority-based fast path architecture for on-chip interconnects. In *Proceedings of IEEE Symposium on High-Performance Interconnects*, pages 15–20, 2007.

[12] W.J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[13] S. A. Felperin, L. Gravano, G. D. Pifarre, and J. L. C. Sanz. Fully-adaptive routing: packet switching performance and wormhole algorithms. In *Proceedings of ACM/IEEE Conference on Supercomputing*, pages 654–663, 1991.

[14] W.J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–475, April 1993.

[15] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[16] Y.M. Boura and C.R. Das. Efficient fully adaptive wormhole routing in n-dimensional meshes. In *Proceedings of International Conference on Distributed Computing Systems*, pages 589–596, June 1994.

[17] J.H. Upadhyay, V. Varavithya, and P. Mohapatra. Efficient and balanced adaptive routing in two-dimensional meshes. In *Proceedings of IEEE Symposium on High-Performance Computer Architecture*, 1995.

[18] J. Flich, S. Rodrigo, and J. Duato. An efficient implementation of distributed routing algorithms for

nocs. In *Proceedings of ACM/IEEE International Symposium on Networks-on-Chip*, pages 87–96, 2008.

[19] L.-S. Peh and W.J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of International Symposium on High-Performance Computer Architecture*, 2001.

[20] E.S. Shin, III Mooney, V.J., and G.F. Riley. Round-robin arbiter design and generation. In *Proceedings of International Symposium on System Synthesis*, 2002.

[21] C.J. Glass and L.M. Ni. The turn model for adaptive routing. In *Proceedings of International Symposium on Computer Architecture*, 1992.

[22] G.M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729–738, July 2000.

[23] L. Zhang, Y. Han, Q. Xu, X. Li, and H. Li. On topology reconfiguration for defect-tolerant noc-based homogeneous manycore systems. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, 17(9):1173–1186, 2009.

[24] T.L. Rodeheffer and M.D. Schroeder. Automatic reconfiguration in autonet. In *Proceedings of ACM symposium on operating systems principles*, pages 183–197, 1991.

[25] O. Lysne, J.M. Montanana, J. Flich, J. Duato, T.M. Pinkston, and T. Skeie. An efficient and deadlock-free network reconfiguration protocol. *IEEE Transactions on Computers*, 57(6):762–779, 2008.

[26] D. Avresky and N. Natchev. Dynamic reconfiguration in computer clusters with irregular topologies in the presence of multiple node and link failures. *IEEE Transactions on Computers*, 54(5):603–615, May 2005.

[27] J. Wu. A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model. *IEEE Transactions on Computers*, 52(9):1154–1169, 2003.

[28] Z. Zhang, A. Greiner, and S. Taktak. A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 441–446, 2008.

[29] B. Fu, Y. Han, H. Li, and X. Li. A new multiple-round dor routing for 2d network-on-chip meshes. In *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing*, pages 276–281, 2009.

[30] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[31] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie. Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In *Proceedings of International Symposium on Parallel and Distributed Processing*, 2006.

[32] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. Application specific routing algorithms for networks on chip. *IEEE Transactions on Parallel and Distributed Systems*, 20(3):316–330, 2009.

[33] Michel A. Kinsy, Myong Hyon Cho, Tina Wen, Edward Suh, Marten van Dijk, and Srinivas Devadas. Application-aware deadlock-free oblivious routing. In *Proceedings of international symposium on Computer architecture*, pages 208–219, New York, NY, USA, 2009. ACM.

[34] J. Cong, C. Liu, and G. Reinman. Aces: Application-specific cycle elimination and splitting for deadlock-free routing on irregular network-on-chip. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 443–448, 2010.

[35] A. Singh, W.J. Dally, A.K. Gupta, and B. Towles. Adaptive channel queue routing on k-ary n-cubes. In *Proceedings of ACM symposium on Parallelism in Algorithms and Architectures*, pages 11–19, 2004.

[36] D. Seo, A. Ali, W. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *Proceedings of international symposium on Computer Architecture*, pages 432–443, 2005.

[37] N. Agarwal, L.S. Peh, and N. Jha. Garnet: A detailed interconnection network model inside a full-system simulation framework. Technical report, TR CE-P08-001, Princeton University, 2007.

[38] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, K.E. Moore, M.D. Hill, and D.A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, 2005.

[39] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, February 2002.

[40] N. Agarwal, Li-Shiuan Peh, and N.K. Jha. In-network snoop ordering (inso): Snoopy coherence on unordered interconnects. In *Proceedings of International Symposium on High Performance Computer Architecture*, pages 67–78, 2009.

[41] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *Proceedings of International Symposium on Computer Architecture*, pages 24–36, June 1995.

[42] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: characterization and architectural implications. In *Proceedings of international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[43] A.B. Kahng, B. Lin, K. Samadi, and R.S. Ramanujam. Trace-driven optimization of networks-on-chip configurations. In *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, pages 437–442, 2010.

[44] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Addressing manufacturing challenges with cost-efficient fault tolerant routing. In *Proceedings of International Symposium on Networks-on-Chip*, pages 25–32, May 2010.

[45] P.-J. Chuang and N.-F. Tzeng. An efficient submesh allocation strategy for mesh computer systems. In *Proceedings of International Conference on Distributed Computing Systems*, pages 256–263, May 1991.